Amendments to the Claims

Please cancel Claims 7, 9-12, and 23-28. Please amend Claims 1, 8, and 13-17. The Claim Listing below will replace all prior versions of the claims in the application:

Claim Listing

1. (Currently Amended) A method of providing a dispatch table that reflects an execution context of computer readable instructions, the method comprising:

determining an execution context for computer readable instructions; [[and]]

responding to a change in the execution context by using [[a]] the dispatch table which reflects the change;

wherein the execution context for the instructions corresponds to the execution context of a thread executing in a data processing system, the thread being a real-time thread that is a schedulable object;

wherein if the thread is a real-time thread, the execution context of the thread is one of: a real-time thread with no scope on a scope stack; a real-time thread with at least one scope on a scope stack; a no-heap real-time thread with no scope on a scope stack; or a no-heap real-time thread with at least one scope on the stack;

wherein each execution context is associated with a corresponding dispatch table; and

wherein determining a change in execution context includes:

determining that the execution context of the thread has changed; and

responding to the change by causing the thread to be associated with a dispatch table reflecting the change.

2. (Original) A method according to Claim 1 wherein the dispatch table includes one or more addresses, each address associated with a starting location for code associated with an instruction to be interpreted.

3. (Original) A method according to Claim 2 wherein using a dispatch table which reflects the change in execution context further includes rewriting one or more entries of the dispatch table so that the rewritten locations point to code capable of handing the change in execution context.

4. (Original) A method according to Claim 2 wherein using a dispatch table which reflects the change in execution context further includes selecting a predefined dispatch table having one or more addresses that point to code capable of handing the change in execution context.

5. (Original) A method according to Claim 4 wherein the predefined dispatch table is selected from a library of predefined dispatch tables, where each dispatch table in the library reflects a specific execution context.

6. (Original) A method according to Claim 1 wherein determining a change in execution context further includes checking one or more memory references.

7. (Canceled).

8. (Currently Amended) A method according to Claim [[7]] 1 wherein the dispatch table reflecting the change in execution context is capable of managing one of the following execution states of the thread:

unreachable state;

state in which reference or assignment rules are not required;

state in which all assignment rules are enforced;

state in which only reference rules are enforced;

state in which reference and assignment rules, except those that refer to a depth of a scope stack, are enforced; or

state in which all reference and assignment rules are checked.

9-12. (Canceled).

13. (Currently Amended) A method according to Claim [[7]] 1 wherein determining the execution context further includes:

checking one or more memory references; or

checking one or more memory assignments.

14. (Currently Amended) A method according to Claim [[12]] 13 wherein the memory references and the memory assignments are [[is]] allocated from at least one of the following: scope memory, heap memory or immortal memory.

15. (Currently Amended) A method according to Claim [[12]] 13 wherein checking memory references further includes detecting a change in execution context when there are at least two references to scoped memory areas on a scope stack.

16. (Currently Amended) A method according to Claim [[7]] 1 further includes storing [[the]] a previous execution context of the thread as reflected in the dispatch table before the change.

17. (Currently Amended) A method according to Claim 15 further includes in response to the thread returning to [[the]] a previous execution context, causing the thread to be associated with the ~~stored~~ dispatch table.

18. (Original) A method according to Claim 1 wherein using a dispatch table reflecting the change in execution context causes the instructions to be interpreted using memory reference checking rules or memory assignment checking rules.

19. (Original) A method according to Claim 18 further including throwing an exception if there is a violation of any of the reference or assignment checking rules.

20. (Original) A method according to Claim 1 wherein the dispatch table is used to implement the instructions into machine executable code.

21. (Original) A method according to Claim 1 wherein a bytecode interpreter causes the dispatch table to reflect the change in execution context.

22. (Original) A method according to Claim 1 wherein the dispatch table is any one of: a bytecode vector table or opcode vector table.

23-28. (Canceled).